DO LANGUAGE MODELS NEED A TRAINABLE INPUT EMBEDDING TABLE? FIXED MINIMAL TOKEN CODES AT 1.7B-CLASS SCALE

Anonymous authors
Paper under double-blind review

ABSTRACT

A trainable input embedding table assigns each vocabulary item an independently adjustable vector. We investigate whether this token-specific parameterization is required for substantial language-modeling capability, or whether a shared Transformer can learn from fixed token identities. We compare three decoder-only language models trained from scratch with the same tokenizer, contextual backbone, untied output-head architecture, and training recipe, with a target budget of 100 billion prediction tokens per model. Their input interfaces are a learned table, canonical 16-bit token-ID codes, and one fixed invertible recoding over GF(2). The fixed codes are repeated to model width without an additional trainable input projection. Both fixed-code models acquire substantial capabilities: canonical codes achieve 52.40% HellaSwag normalized accuracy, 70.51% PIQA accuracy, and 42.75% LAMBADA accuracy. The learned-input control performs better on several evaluations, including HellaSwag and LAMBADA, so these results establish viability rather than performance parity. The fixed interfaces remove 100.7 million trainable parameters, yielding 1.711B-parameter models, but parameter reduction is not the central result. These single-run experiments distinguish architectural necessity from empirical utility: independently trainable token-specific input vectors are not required for the observed capabilities. A fixed identity interface also provides a controlled setting for studying representation learning downstream of an immutable input, without establishing where particular capabilities are localized.

1 INTRODUCTION

A language model must distinguish its input symbols, but it need not represent that distinction through an independently trainable vector for every symbol. Conventional neural language models combine these two choices: each token identifies a row of an embedding table, and that row is optimized together with the rest of the network (Bengio et al., 2003; Vaswani et al., 2017). The resulting interface is effective and widely used. Its success, however, does not establish that this particular allocation of trainable parameters is necessary.

We separate three questions:

1. **Identity:** does the input interface preserve which token was observed?
2. **Learnability:** can a shared network learn useful language computations from that interface?
3. **Utility:** does independently adapting the input vector of each token improve the resulting model?

The first is a coding question. The second and third require empirical evidence. An injective code answers the first question but does not, by itself, answer either of the others.

This distinction matters beyond embedding compression. With a learned table, both the input coordinates and the computation acting on them change during training. With a fixed interface, the coordinates of every token remain unchanged: all learned adaptation occurs in the remaining trainable system. This creates a controlled architectural restriction under which to investigate the construction

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of lexical and contextual representations. It does not make those representations automatically interpretable, but it removes one source of representational change.

We study an intentionally simple restriction. For a vocabulary of size V, a fixed-length binary identity code requires

$$K = \lceil \log_2 V \rceil$$

bits. For our vocabulary of 49,152 tokens, $K = 16$. We repeat these coordinates to the Transformer width without adding a trainable projection. The model therefore receives exact token identity through a fixed, low-dimensional interface rather than a learned token-specific row. A second fixed-code model uses an invertible linear recoding over $\mathrm{GF}(2)$.

The experiment is not motivated by a claim that binary coordinates are uniquely suitable for language. They provide a transparent, collision-free construction whose information content and trainable parameter count are easy to specify. The object of study is the separation of fixed token identity from learned use of that identity.

We train three models with a common contemporary decoder backbone and a target budget of 100B prediction tokens per model. Unlike evidence based only on training convergence or fluent examples, our evaluation includes standard base-model multiple-choice tasks, last-word prediction, and corpus likelihood. The fixed-code models exhibit substantial capability across several of these evaluations. They nevertheless underperform the learned-input control on important tasks, including HellaSwag and LAMBADA.

The central finding is therefore a distinction, not a dominance claim: *the learned input table is useful, but it is not required for the capabilities demonstrated here*. Removing it changes the conditions under which representations are learned; it does not eliminate representation learning.

Contributions. We provide:

- a backbone-matched comparison of learned and fixed token-identity interfaces at 1.7B-class scale, using a standard tokenizer and broad base-model evaluation;
- a measurement of both the capabilities retained and the quality cost incurred when all independently trainable token-specific input vectors are removed;
- a noncanonical fixed-code experiment demonstrating viability under one invertible recoding, without claiming invariance to code assignment;
- an explicit account of what an immutable input interface controls, what it does not identify, and how it can support subsequent studies of representation learning.

2 Related Work

Learned and shared lexical parameterizations. Trainable token vectors are a standard component of neural language models (Bengio et al., 2003; Vaswani et al., 2017). Weight tying shares the input and output vocabulary matrices (Press & Wolf, 2017), while factorization reduces the size of the learned lexical interface (Lan et al., 2020). These methods demonstrate that parameter allocation at the vocabulary boundary is already an architectural choice rather than a single fixed design. Our parameter counts are relative to an untied baseline; they should not be interpreted as additional savings over a tied model.

Compositional and generated embeddings. Compositional code learning represents words through combinations of shared codeword vectors (Shu & Nakayama, 2018). ALONE constructs word representations from fixed word-specific filters, a shared base embedding, and a trainable feed-forward network (Takase & Kobayashi, 2020). Subword-based compact reconstruction uses shared representations and subword information to reconstruct lexical vectors (Sasaki et al., 2021). Hash embeddings likewise replace independent vocabulary rows with parameter sharing (Svenstrup et al., 2017).

These are substantive precedents. They already challenge the need for an independently stored high-dimensional vector for every token. We do not claim that shared lexical computation, many-hot codes, or embedding generation are new. Our experiment studies a restrictive operating point within

this broader design space: fixed minimal binary identities, a fixed lift, and no additional learned input generator before the standard Transformer backbone.

Frozen inputs and bit-token interfaces. Frozen visual Unicode representations and fixed token-identity ablations have been investigated in Transformer language models (Bochkov, 2025). MaskBit demonstrates embedding-free generation through bit tokens in the image domain (Weber et al., 2024). It is an important precedent for direct bit-token interfaces, although its modality, token construction, and generation setting differ from the causal language-model experiment studied here.

Our contribution is not the first use of binary representations or the first successful model without a conventional input embedding table. It is a controlled empirical characterization of a particularly restricted input interface at the present scale, including standard language-model evaluations and the observed cost relative to learned inputs.

Tokenization versus input parameterization. Character-aware models and byte-level models change the units from which representations are constructed (Kim et al., 2016; Xue et al., 2022). Here tokenization is held fixed. We change the map from existing token IDs to the continuous input of the contextual network, not the vocabulary or the sequence of tokens presented to the model.

External quality references. SmolLM2 provides publicly available base models at several sizes (Ben Allal et al., 2025). We evaluate them with the same local benchmark protocol to calibrate the absolute quality of our models. They are not matched training controls. The causal comparison of interest is between the three models trained within our study.

3 Fixed Token-Identity Interfaces

3.1 Learned input and output parameterizations

Let a token ID be $t \in \{0, \dots, V-1\}$. The learned-input control uses

$$x(t) = E_{\text{in}}[t], \qquad E_{\text{in}} \in \mathbb{R}^{V \times d}.$$

Each row is independently adjustable during optimization.

All models retain an untied, trainable output projection:

$$h_i = F_\theta(x(t_1), \dots, x(t_i)), \qquad p(t_{i+1} \mid t_{\le i}) = \text{softmax}(W_{\text{out}} h_i),$$

where

$$W_{\text{out}} \in \mathbb{R}^{V \times d}.$$

Thus, eliminating trainable input rows does not eliminate vocabulary-specific trainable parameters from the complete model.

3.2 Canonical binary identities

We assign each token its little-endian binary expansion:

$$c(t) \in \{0,1\}^K, \qquad c(t)_j = \left\lfloor \frac{t}{2^j} \right\rfloor \bmod 2, \qquad j \in \{0, \dots, K-1\}.$$

For the vocabulary used here,

$$V = 49{,}152, \qquad K = \lceil \log_2 V \rceil = 16.$$

The term *minimal* refers only to the width of an injective fixed-length binary code. It does not mean minimum real-valued dimension, minimum entropy-coded description length, or minimum total model storage.

Token identity is a categorical variable. The code supplies a vector-valued representation of that variable; its individual bits are not assumed to correspond to linguistic attributes. Nor do we assume that neighboring codes represent semantically related tokens.

3.3 Deterministic lift

The model width is $d = 2048$. We repeat the code 128 times:

$$x(t) = Rc(t), \qquad R = \begin{bmatrix} I_{16} \\ I_{16} \\ \vdots \\ I_{16} \end{bmatrix} \in \mathbb{R}^{2048 \times 16}.$$

This lift has full column rank and no trainable parameters. The code values are zero and one, with scale one.

No learned input projection, codeword vectors, or auxiliary embedding generator is inserted before the standard backbone. The backbone itself remains trainable, including its initial normalization and attention transformations.

3.4 Fixed invertible recoding

The third model replaces the canonical assignment by

$$\widetilde{c}(t) = Ac(t) \pmod 2, \qquad A \in \mathrm{GL}(16, 2),$$

and uses

$$x(t) = R\widetilde{c}(t).$$

The matrix is generated once with code seed 12345. Generation requires full rank over $\mathrm{GF}(2)$ and a minimum Hamming weight of four in every row and column. The implementation supports an affine offset, but the evaluated configuration sets $b = 0$. The transformation studied here is therefore linear over $\mathrm{GF}(2)$, not a randomly shifted affine map.

Invertibility preserves token identity. It does not preserve Hamming distance, real-valued input norms, or optimization behavior. Moreover, $V < 2^{16}$: the vocabulary occupies only a subset of the binary hypercube. Full-hypercube balance properties therefore cannot automatically be attributed to the instantiated vocabulary.

Only one recoding matrix is evaluated. The experiment asks whether useful learning survives this particular change in assignment, not whether all assignments are equivalent.

3.5 Implementation and scope

Both fixed-code implementations store a persistent $V \times 16$ codebook buffer. The GF2 implementation additionally stores its matrix and offset. These are fixed stored values, not trainable parameters.

The codebooks could be regenerated algorithmically, but the evaluated implementations perform a lookup into a compact buffer. We therefore describe the models as having *no trainable input table*, not as literally lookup-free implementations.

The tokenizer itself is corpus-derived, and token enumeration may carry incidental structure. Our restriction removes optimization of the token-level input coordinates during model training; it does not remove every possible source of corpus information from the input pipeline.

4 Experimental Design

4.1 Models and controlled comparison

All three models are trained from scratch. They use the same tokenizer family, 24 decoder blocks, hidden width 2048, 32 attention heads, RoPE, RMSNorm, and SwiGLU. The feed-forward intermediate width is 8192. Training context length is 2048 tokens. Attention and feed-forward projections are bias-free, dropout is zero, and the output vocabulary projection is untied.

The models share the contextual architecture, output-head architecture, preprocessing, sampling procedure, and optimization recipe. The input parameterization changes. We do not reallocate the removed input parameters into the backbone or independently tune the recipe for each interface.

Table 1: Parameter accounting. The body excludes the input interface and output projection. Input-buffer entries are stored tensor values, not trainable parameters and not bytes. The fixed interfaces remove 100.7M trainable parameters, approximately 5.56% of the untied learned-input model.

Quantity	LEARNED	BINARY16	GF2
Trainable input parameters	100,663,296	0	0
Trainable body parameters	1,610,713,088	1,610,713,088	1,610,713,088
Trainable output parameters	100,663,296	100,663,296	100,663,296
Total trainable parameters	1,812,039,680	1,711,376,384	1,711,376,384
Persistent input-buffer entries	0	786,432	786,704

This is a *backbone-matched component intervention*. It measures what happens when a particular input parameterization and its associated capacity are removed. It is not a comparison of optimal models at equal parameter count, equal measured FLOPs, or equal wall-clock time.

4.2 DATA AND SAMPLING

We use the `fineweb-edu-dedup` component of `HuggingFaceTB/smollm-corpus`, derived from the FineWeb-Edu data family (Penedo et al., 2024). The preprocessing configuration processes all 234 source Parquet shards. No Cosmopedia mixture is used in the reported configuration.

Training samples remain within document boundaries. At context length 2048, only documents containing at least 2049 tokens are eligible. Within the resident shard, an eligible document is sampled uniformly and then a valid window start is sampled uniformly. Each shard visit supplies 1024 microbatches per rank before the sampler advances. Training and validation documents are partitioned by a seeded hash of the document identifier, with a text-hash fallback and a validation threshold of 0.001. This procedure samples with replacement and is not uniform over all corpus tokens. Reported training-token budgets refer to processed prediction targets, including repeated or overlapping windows, rather than distinct corpus tokens.

4.3 OPTIMIZATION AND REPLICATION

Each model has a target budget of 100B prediction tokens. The launch configuration uses two GPUs, eight sequences per GPU, and eight gradient-accumulation steps. The global number of prediction targets per optimizer update is

$$2 \times 8 \times 8 \times 2048 = 262{,}144.$$

We use AdamW with peak learning rate 1.5×10^{-4}, minimum scheduled learning rate 10^{-5}, 2000 warmup updates, betas $(0.9, 0.95)$, and epsilon 10^{-8}. Weight decay is 0.01 on matrix-valued parameters and zero on one-dimensional parameters. The learning rate follows linear warmup and cosine decay over a configured maximum of 382,500 updates. Gradient-norm clipping is 1.0.

Model parameters remain FP32 during training, with BF16 autocast for eligible operations and gradient checkpointing. Training stops at an optimizer-step boundary when the token threshold is reached. Actual checkpoint counters, rather than launch arguments, determine the realized budget.

There is one training run per input interface. Evaluation standard errors do not estimate training-seed variability. The released trainer also does not restore the full per-rank sampler cursor and dedicated sampling-generator state on resume. Consequently, a shared recipe and nominal seed do not establish identical realized sample order across interrupted runs.

Architecture and optimization details appear in Appendix B. Links to the released artifacts appear in Appendix E.

4.4 EVALUATION

We evaluate base models without a chat template using the Language Model Evaluation Harness (EleutherAI). The suite comprises HellaSwag (Zellers et al., 2019), ARC (Clark et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), OpenBookQA (Mihaylov et al., 2018),

Table 2: Selected base-model evaluations. Accuracy values and their standard errors are in percentage points. Each column represents one training run. The fixed-code models retain substantial capability but do not match the learned-input control across the suite. Complete results appear in Appendix A.

Metric	LEARNED	BINARY16	GF2
HellaSwag, normalized accuracy	57.79 ± 0.49	52.40 ± 0.50	51.44 ± 0.50
ARC-Easy, normalized accuracy	66.04 ± 0.97	61.53 ± 1.00	60.73 ± 1.00
ARC-Challenge, normalized accuracy	37.63 ± 1.42	34.04 ± 1.38	34.22 ± 1.39
PIQA, accuracy	72.69 ± 1.04	70.51 ± 1.06	71.16 ± 1.06
WinoGrande, accuracy	58.56 ± 1.38	55.33 ± 1.40	55.01 ± 1.40
OpenBookQA, normalized accuracy	37.80 ± 2.17	38.00 ± 2.17	36.80 ± 2.16
LAMBADA, accuracy	47.72 ± 0.70	42.75 ± 0.69	42.29 ± 0.69
WikiText, word perplexity ↓	16.50	18.58	19.03

CommonsenseQA (Talmor et al., 2019), MMLU (Hendrycks et al., 2021), LAMBADA (Paperno et al., 2016), and WikiText (Merity et al., 2017).

Multiple-choice tasks are evaluated zero-shot, with MMLU also reported at five shots. We preserve the distinction between task-defined accuracy and length-normalized accuracy. LAMBADA perplexity scores the target continuation rather than whole documents. WikiText word perplexity and bits per byte use the harness's task-specific normalization.

The main table is a compact selection of labeled metrics; the complete score table is reported in Appendix A. No aggregate capability score or post hoc claim of a preregistered primary metric is made.

The evaluation audit records identical sample/prompt multisets across the six models within each task group. For five-shot MMLU, 1508 of 56,168 candidate likelihood requests are marked as truncated for each model. These are candidate requests, not necessarily distinct questions. The result is labeled accordingly.

Reported $\pm$ values are evaluation standard errors, not standard deviations across training runs. We do not infer paired significance from marginal error bars. No paired test or training-seed confidence interval is reported.

4.5 EXTERNAL REFERENCES

We evaluate the base checkpoints SmolLM2-135M, SmolLM2-360M, and SmolLM2-1.7B using the same local protocol. Their reported training budgets are approximately 2T, 4T, and 11T tokens, respectively (Ben Allal et al., 2025).

These models differ from ours in size, architecture, data composition, schedule, and compute. They provide external quality references, not controls for sample efficiency or the effect of the input interface. In particular, their larger token budgets must not be converted into an efficiency claim for fixed codes.

5 RESULTS

5.1 FIXED IDENTITIES SUPPORT SUBSTANTIAL CAPABILITY

The canonical fixed-code model reaches 52.40% HellaSwag normalized accuracy, 70.51% PIQA accuracy, and 42.75% LAMBADA accuracy. The corresponding GF2 values are 51.44%, 71.16%, and 42.29%. Both also achieve useful corpus likelihood on WikiText.

These observations go beyond training convergence or isolated generation examples. HellaSwag and PIQA performance is well above uniform-choice reference levels, while LAMBADA and WikiText demonstrate nontrivial next-token prediction in different evaluation formats.

Uniform guessing is not a complete benchmark baseline, and these results do not establish general reasoning. Nevertheless, the combination of multiple-choice performance, exact last-word prediction,

and corpus likelihood demonstrates that the fixed interface supports meaningful language-modeling behavior.

5.2 A NONCANONICAL ASSIGNMENT REMAINS VIABLE

The recoded model performs in the same broad range as the canonical fixed-code model. Canonical codes have better HellaSwag and WikiText results, whereas the ordering reverses on some other metrics.

This demonstrates viability under one noncanonical assignment. It does not establish invariance to input geometry or a distributional robustness result over recoding matrices. The transformation changes the association between tokens, bit patterns, norms, and neighborhoods. One matrix cannot separate these effects.

5.3 NEAR-CHANCE TASKS DELIMIT THE EVIDENCE

CommonsenseQA accuracy is approximately 20% for all three models. MMLU scores remain approximately 25–26% at both shot counts. These values are close to uniform-choice reference levels.

We report these outcomes rather than reinterpret them as evidence of broad knowledge or reasoning. They limit the scope of the result: fixed codes support the capabilities demonstrated on informative tasks, not every capability one might seek from a language model.

5.4 EXTERNAL REFERENCES CALIBRATE ABSOLUTE QUALITY

The fixed-code models exceed SmolLM2-135M on several metrics, including HellaSwag and WikiText, but trail SmolLM2-360M on several others. For example, canonical-code HellaSwag normalized accuracy is 52.40%, between SmolLM2-135M at 43.02% and SmolLM2-360M at 56.28%.

SmolLM2-1.7B is substantially stronger on the reported suite. The fixed-code models are therefore meaningful pretrained language models, not competitive replacements for a much more extensively trained model of similar size. The external comparisons do not explain the internal learned-versus-fixed gap.

6 WHAT THE EXPERIMENT ESTABLISHES

6.1 NECESSITY AND UTILITY ARE DIFFERENT CLAIMS

A component may improve a model without being required for the behavior under investigation. The present results illustrate this distinction: the learned input table benefits several evaluations, yet the fixed-code models retain substantial capability.

A constructive example can establish feasibility. It cannot establish universal optimality, equivalence under all training conditions, or the irrelevance of the removed component. Our evidence is therefore an operating-point result: a specific family of fixed identity interfaces works at the tested scale and budget, with an observed quality cost.

There is no established impossibility theorem being overturned. Previous shared and generated embeddings already motivate alternatives to independent vocabulary rows. The contribution is empirical evidence about a restrictive interface, not a claim that standard theory predicts failure.

6.2 REMOVING A TABLE DOES NOT REMOVE AN EMBEDDING FUNCTION

After the fixed lift, trainable shared transformations act on the token codes. These can learn context-independent lexical functions as well as contextual computations. In a broad functional sense, they may be described as constructing embeddings.

That description does not make them the same parameterization as an independently trainable lookup table. A learned lookup can adjust a token's row directly. A fixed-code model changes token representations through shared parameters acting on immutable coordinates.

For a linear transformation applied directly to the raw input,

$$Wx(t) = WRc(t).$$

This makes the shared structure explicit. In the actual network, RMSNorm precedes attention, and later operations include nonlinearities, attention, and residual connections. The raw input-rank restriction must not be extended to all subsequent hidden states.

The scientifically relevant distinction is therefore *independently adjustable input rows versus shared learned computation over fixed identities*, not learned versus unlearned representations throughout the model.

6.3 An immutable input controls a boundary, not an entire system

In the fixed-code runs, the input vector associated with a given token is exactly the same throughout training. If behavior changes, that change cannot be attributed to movement of the token's input coordinates.

This provides a useful experimental boundary condition. It does not make hidden representations unique, align independently trained networks, or remove internal reparameterization freedoms. Two models with the same fixed input can learn very different internal organizations.

Similarly, the experiment does not show that semantic information was transferred from an existing learned table into particular layers. The models are trained from scratch. The defensible conclusion is that useful learned behavior can be constructed without adapting that table in the first place.

The first trainable layer is one possible site of lexical computation, not a demonstrated exclusive location of it. The present results neither localize a capability to that layer nor establish that it is distributed across all layers. The output projection also remains trainable and vocabulary-specific.

6.4 Binary minimality is an experimental choice

An input representation must distinguish tokens, but injectivity alone does not guarantee easy optimization. The binary interface imposes a particular geometry and low-dimensional structure that may help or hinder finite-budget learning.

Other fixed encodings, including random continuous codes and suitably constructed sinusoidal codes, could investigate the same broader question. Their collision behavior, numerical precision, scale, and learnability would require separate checks. The current results do not establish that they would perform equally well.

The significance of the binary construction is its clarity: identity preservation and absence of trainable input coordinates are explicit. The number sixteen is a consequence of the vocabulary size, not a proposed universal dimension for language.

7 Why a Fixed Interface Is a Useful Research Instrument

The value of this intervention is not exhausted by parameter accounting. It creates a viable model family in which token identity is held fixed while learned computation changes. This supports several concrete lines of investigation.

Representation formation without a moving input. One can track how lexical and contextual distinctions become accessible across training and depth when the raw token coordinates remain unchanged. Such studies should combine representational measurements with causal interventions, rather than infer a mechanism from visual clustering alone. Layer ablations, controlled activation substitutions, and tests on matched contexts could distinguish accessible information from information used for a prediction.

Separating code geometry from trainability. Learned low-dimensional coordinates, fixed random continuous codes, norm-matched controls, and multiple independently sampled recodings would test different parts of the current intervention. These comparisons could determine whether the observed

cost is mainly associated with fixedness, dimensional restriction, input scale, or code assignment. They would explain why a viable interface works rather than merely demonstrate that it can work.

Stable boundaries for modular systems. A reproducible token-identity interface could serve as one boundary in systems whose contextual computation, memories, or executable components are constructed and updated separately. The attraction is operational: an input specification can remain unchanged while other components are replaced or studied.

These directions treat the fixed interface as an experimental instrument. They do not require it to be the best production parameterization, and they do not turn the present benchmark results into a mechanistic account of reasoning.

8 LIMITATIONS

One training run per interface. The study does not estimate variability across training initializations or realized data streams. Task-level standard errors cannot substitute for independent training runs. The reported differences describe these checkpoints rather than a fully estimated population of training outcomes.

A complete parameterization intervention. Input trainability, parameter count, geometry, and scale change together. The fixed zero/one vectors have different initial norms from the randomly initialized learned vectors. The shared recipe was not independently optimized for each interface. Accordingly, the quality gap cannot be attributed to a single mechanism.

Restricted generalization. The experiment uses one tokenizer family, vocabulary size, context length, data configuration, and main model scale. It does not establish behavior in multilingual settings, long-context operation, instruction tuning, or substantially different training budgets. Near-chance MMLU and CommonsenseQA scores particularly limit claims about broad reasoning and knowledge.

No demonstrated systems-efficiency advantage. Removing input parameters also removes their optimizer state, but runtime depends on buffer access, tiling, activations, and kernel behavior. We report no measured throughput, energy, or latency advantage. For a baseline with tied input and output weights, removing the input use of a matrix need not remove the matrix from storage.

The trainable output interface remains. The output head retains a full vocabulary-sized trainable matrix. The result concerns input parameterization, not elimination of all learned lexical storage.

9 CONCLUSION

Fixed minimal token identities support substantial language-modeling capability at 1.7B-class scale without independently trainable token-specific input vectors. This holds for canonical binary codes and for one fixed invertible recoding. The learned-input control nevertheless performs better on several informative evaluations.

The result separates two questions that are often treated together: whether a learned input table is useful, and whether it is required. In the tested setting it is useful, but not required for the capabilities observed.

Beyond the removed parameters, the experiment supplies a controlled setting in which input identity is immutable and representation learning proceeds in the remaining trainable system. This makes fixed interfaces a useful starting point for studying how capabilities are constructed and how learned components might be separated, without claiming that either their mechanisms or their locations have already been identified.

REPRODUCIBILITY STATEMENT

Anonymized research artifacts are available at https://huggingface.co/E6E831728. It includes the original training implementations, model configurations, and checkpoint-loading utilities for LEARNED, BINARY16, and GF2.

The input-verification utility checks parameter counts, the complete fixed codebooks, the rank and offset of the GF2 transformation, and the deterministic lift. It also checks loading, finite forward logits, and a short generation. These checks establish specific implementation properties; they are not benchmark evaluations or a complete proof of equivalence between training and inference implementations.

The original trainer uses already-shifted target labels. An exported inference adapter may instead use the standard Transformers label convention. Reproduction must respect this distinction. Logit comparison is independent of that label convention.

The released model checkpoints and loading utilities are listed in Appendix E. Data preparation and training source code are provided in the supplementary material.

ETHICS STATEMENT

The primary intended benefit is a clearer experimental separation between token identity and learned language computation. The resulting models retain the ordinary limitations and risks of pretrained language models, including biased outputs, unsupported factual statements, and potential misuse.

A fixed input interface is not a safety mechanism. It does not establish factual reliability, interpretability of internal reasoning, or control over generated content. The released checkpoints are base research models rather than validated instruction-following assistants.

AI USE STATEMENT

Generative AI tools were used for research discussion, literature discovery, assistance with interpreting experimental results, and drafting and editing the manuscript and documentation. The authors reviewed the AI-assisted material and are responsible for the experiments, code, references, and claims.

REFERENCES

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big—data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. URL https://www.jmlr.org/papers/v3/bengio03a.html.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Andrey Bochkov. Emergent semantics beyond token embeddings: Transformer LMs with frozen visual unicode representations. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Odh8IynO1o.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

EleutherAI. Language model evaluation harness. Open-source software. URL https://github.com/EleutherAI/lm-evaluation-harness.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.03300.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016. URL https://arxiv.org/abs/1508.06615.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1909.11942.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.07843.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://aclanthology.org/D18-1260/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, 2016. URL https://aclanthology.org/P16-1144/.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.17557.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 157–163, 2017. URL https://aclanthology.org/E17-2025/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1907.10641.

Shota Sasaki, Jun Suzuki, and Kentaro Inui. Subword-based compact reconstruction for open-vocabulary neural word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3551–3564, 2021. doi: 10.1109/TASLP.2021.3125133. URL https://doi.org/10.1109/TASLP.2021.3125133.

Raphael Shu and Hideki Nakayama. Compressing word embeddings via deep compositional code learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJRZzFlRb.

Dan Svenstrup, Jonas Hansen, and Ole Winther. Hash embeddings for efficient word representations. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1709.03933.

Sho Takase and Sosuke Kobayashi. All word embeddings from one embedding. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/275d7fb2fd45098ad5c3ece2ed4a2824-Abstract.html.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 2019. URL https://aclanthology.org/N19-1421/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

Mark Weber, Lijun Yu, Qihang Yu, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. MaskBit: Embedding-free image generation via bit tokens. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2409.16211.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022. URL https://arxiv.org/abs/2105.13626.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://aclanthology.org/P19-1472/.

A Complete Evaluation Results

Table 3 reports the complete consolidated score table. The $\pm$ entries are evaluation standard errors. They do not quantify training-run uncertainty.

The three internal models share the experimental design described in Section 4. The SmolLM2 checkpoints are external references with different training histories.

Table 3: Complete reported evaluations. Accuracy values and their errors are in percentage points. Perplexities and bits per byte are not percentages. Five-shot MMLU includes context-truncated candidate requests.

Metric	Learned	Binary16	GF2	SmolLM2-135M	SmolLM2-360M	SmolLM2-1.7B
HellaSwag acc	44.21 ± 0.50	40.70 ± 0.49	40.24 ± 0.49	35.36 ± 0.48	43.05 ± 0.49	53.38 ± 0.50
HellaSwag acc_norm	57.79 ± 0.49	52.40 ± 0.50	51.44 ± 0.50	43.02 ± 0.49	56.28 ± 0.50	71.43 ± 0.45
ARC-Easy acc	71.63 ± 0.92	67.59 ± 0.96	66.84 ± 0.97	64.44 ± 0.98	70.24 ± 0.94	77.86 ± 0.85
ARC-Easy acc_norm	66.04 ± 0.97	61.53 ± 1.00	60.73 ± 1.00	58.75 ± 1.01	68.18 ± 0.96	73.36 ± 0.91
ARC-Challenge acc	35.92 ± 1.40	32.34 ± 1.37	30.55 ± 1.35	28.07 ± 1.31	36.26 ± 1.40	44.37 ± 1.45
ARC-Challenge acc_norm	37.63 ± 1.42	34.04 ± 1.38	34.22 ± 1.39	29.61 ± 1.33	38.05 ± 1.42	47.27 ± 1.46
PIQA acc	72.69 ± 1.04	70.51 ± 1.06	71.16 ± 1.06	68.44 ± 1.08	71.38 ± 1.05	76.99 ± 0.98
PIQA acc_norm	72.14 ± 1.05	71.11 ± 1.06	72.14 ± 1.05	68.39 ± 1.08	71.82 ± 1.05	77.20 ± 0.98
WinoGrande acc	58.56 ± 1.38	55.33 ± 1.40	55.01 ± 1.40	52.57 ± 1.40	59.35 ± 1.38	65.98 ± 1.33
OpenBookQA acc	27.60 ± 2.00	28.20 ± 2.01	25.20 ± 1.94	22.00 ± 1.85	24.80 ± 1.93	32.20 ± 2.09
OpenBookQA acc_norm	37.80 ± 2.17	38.00 ± 2.17	36.80 ± 2.16	32.60 ± 2.10	37.80 ± 2.17	44.40 ± 2.22
CommonsenseQA acc	19.82 ± 1.14	20.56 ± 1.16	19.74 ± 1.14	19.90 ± 1.14	21.05 ± 1.17	41.69 ± 1.41
MMLU 0-shot	25.32 ± 0.37	25.88 ± 0.37	26.11 ± 0.37	24.25 ± 0.36	25.47 ± 0.37	48.40 ± 0.41
MMLU 5-shot, partial truncation	25.48 ± 0.37	25.57 ± 0.37	24.66 ± 0.36	25.15 ± 0.36	25.03 ± 0.37	50.06 ± 0.41
LAMBADA accuracy	47.72 ± 0.70	42.75 ± 0.69	42.29 ± 0.69	42.97 ± 0.69	53.31 ± 0.70	67.51 ± 0.65
LAMBADA perplexity	12.88 ± 0.42	17.91 ± 0.62	18.47 ± 0.63	19.06 ± 0.63	9.38 ± 0.27	4.44 ± 0.10
WikiText word perplexity	16.50	18.58	19.03	23.14	17.12	11.62
WikiText byte perplexity	1.69	1.73	1.73	1.80	1.70	1.58
WikiText bits/byte	0.76	0.79	0.79	0.85	0.77	0.66

WikiText byte perplexity and bits per byte are two parameterizations of the same byte-normalized likelihood, rather than independent pieces of evidence. Word perplexity uses a different normalization of the likelihood. Likewise, accuracy and normalized accuracy on the same task should not be counted as independent benchmark replications.

B Architecture and Training Details

Initialization. Linear layers and the learned input table use normally distributed initialization with standard deviation 0.02. Attention output projections and feed-forward down projections are

Table 4: Configured architecture and optimization. The target budget and schedule length do not substitute for the realized counters of the evaluated checkpoints.

Setting	Value
Vocabulary size	49,152
Decoder blocks	24
Hidden width	2048
Attention heads	32
Head dimension	64
SwiGLU intermediate width	8192
Context length	2048
Positional encoding	RoPE
RoPE base	10,000
Normalization	RMSNorm
RMSNorm epsilon	10^{-5}
Attention/MLP projection biases	Disabled
Dropout	0
Output head	Trainable, untied, bias-free
Optimizer	AdamW
Peak learning rate	1.5×10^{-4}
Minimum scheduled learning rate	10^{-5}
Warmup	2000 updates
Configured schedule length	382,500 updates
Betas	$(0.9, 0.95)$
Optimizer epsilon	10^{-8}
Matrix-parameter weight decay	0.01
Other-parameter weight decay	0
Gradient clipping	1.0
GPUs per run	2
Sequences per GPU per microbatch	8
Gradient accumulation	8
Prediction targets per update	262,144
Target prediction-token budget	10^{11}
Parameter dtype during training	FP32
Autocast dtype	BF16
Gradient checkpointing	Enabled

initialized with standard deviation

$$\frac{0.02}{\sqrt{2L}},$$

where $L = 24$ is the number of blocks. The fixed-code models replace the input module while retaining the same backbone construction procedure.

The initial input scales are not matched. For a fixed token with Hamming weight h, the squared norm of the raw tiled zero/one input is

$$\|x(t)\|_2^2 = 128h.$$

RMSNorm affects the attention input, but the raw representation also participates in the residual stream. Consequently, normalization does not make input scaling an irrelevant implementation detail.

Target alignment. The training sampler returns sequences of length 2049. Inputs are the first 2048 tokens, and targets are the last 2048 tokens. The original model implementation computes cross-entropy against these already-shifted targets without shifting them again. All 2048 positions contribute prediction targets.

Data weighting. The sampler selects documents uniformly within a resident shard rather than weighting them by length. It then samples a window within the selected document. Documents shorter than 2049 tokens do not enter either training-window sampling or the corresponding in-training validation-window sampling.

Uniform document sampling, fixed microbatches per shard, and sampling with replacement determine the effective training distribution. It should not be described as one pass through all corpus tokens.

Tokenizer sources. The preprocessing command names `HuggingFaceTB/SmolLM2-1.7B` as its tokenizer source. These repositories are intended to provide the common SmolLM2 tokenizer.

C Evaluation Audit and Interpretation

Table 5: Reported request-level audit for each of the six models. Counts refer to the audit's request units, not uniformly to benchmark questions or internal model windows.

Task group	Likelihood requests	Marked truncated
Multiple choice, 0-shot	68,671	0
MMLU, 0-shot	56,168	0
MMLU, 5-shot	56,168	1,508
LAMBADA	5,153	0
WikiText rolling likelihood	173	0

Matching sample/prompt multisets supports consistency of the evaluated text across models. It does not independently validate likelihood alignment, special-token handling, every adapter path, or uncertainty aggregation.

For five-shot MMLU, the fraction of candidate requests marked as truncated is approximately 2.68%. The score is retained as an explicitly qualified result. It should not be presented as an unrestricted full-prompt five-shot evaluation.

D Elementary Properties and Boundary Cases

Minimal fixed-length binary width. There are 2^K binary vectors of width K. An injective representation of V symbols therefore requires $2^K \geq V$, implying

$$K \geq \lceil \log_2 V \rceil.$$

The canonical construction attains this bound. This is a coding fact, not a language-model learnability result.

Injectivity. Distinct token IDs have distinct canonical binary codes. Multiplication by an invertible matrix over $\mathrm{GF}(2)$ preserves that distinction. The real-valued lift R has full column rank, so it also preserves distinct code vectors. Thus both fixed interfaces are collision-free on the vocabulary.

Raw input rank. All raw input vectors lie in the column space of R:

$$\dim \operatorname{span}\{x(t) : t \in \mathcal{V}\} \leq 16.$$

This is a restriction on the input representation, not on the rank of every later activation matrix. Normalization, nonlinearities, attention, and residual computation act after the raw input.

Zero-code boundary case. Canonical zero/one coding maps token ID zero to the zero vector. The zero-offset GF2 recoding preserves it.

In the supplied bias-free architecture, an input sequence consisting entirely of zero-code tokens yields zero hidden states and zero output logits, hence a uniform next-token distribution, assuming finite parameters.

A zero-code token following a nonzero context can receive information through causal attention. This boundary case therefore does not imply that every occurrence of token zero is unusable. It does require documenting how empty contexts and initial special tokens are represented during evaluation.

Fixed input does not identify internal coordinates. An immutable token map constrains the observable input interface. It does not uniquely determine the learned parameters or the internal basis used by hidden representations. Cross-model activation comparison still requires appropriate controls or alignment. Modular compatibility is not guaranteed by shared token coordinates alone.

E ARTIFACTS

The anonymous release location is `https://huggingface.co/E6E831728`, with anonymized repositories for the three evaluated interfaces:

- `E6E831728/ab_ext_learned`;
- `E6E831728/ab_ext_binary16`;
- `E6E831728/ab_ext_gf2`.

All dataset preparation and training scripts, including model source code, are available in the Supplementary Material.